# Developing Zynq Software with Xilinx SDK

## Lab 2

## Importing the Hardware Platform into SDK


August 2016
Version 09

## Lab 2 Overview

The Xilinx® Software Development Kit (SDK) is a full-fledged embedded software development environment for Xilinx embedded processors. SDK is based on the Eclipse open source platform. SDK features include the following:

- Feature-rich C/C++ code editor and compilation environment
- Project management
- Application build configuration and automatic Makefile generation
- Error navigation
- Well-integrated environment for source-level debugging and profiling of embedded targets
- Project archiving

In addition to the above native Eclipse provided features, SDK also provides the following tools for use in Xilinx embedded software development:

- Reference software applications
- Xilinx Microprocessor Debugger (XMD): A debug agent used to communicate with Xilinx embedded processors using JTAG. It also provides various low-level debugging features not directly available in SDK.
- SoC programmer: Used to program the Xilinx SoC with the bitstream.
- Flash programmer: Used for burning bitstreams and software application images into external Flash devices
- FSBL generator: Used for automatically bootloading your embedded software applications from Flash
- Linker script generator: Used for quickly mapping your application image across the hardware memory space

## Lab 2 Objectives

When you have completed Lab 2, you will know how to do the following:

- Set up an SDK workspace
- Import a pre-built Zynq hardware platform.
- Briefly examine the hardware platform including documentation for each peripheral.

# Experiment 1: Workspace and Hardware Platform

This experiment shows how to set up a workspace and import a hardware platform.

***Workspaces***

SDK uses the concept of workspaces to hold your software development work. A workspace is a directory in your file system that SDK uses to hold meta-information about the projects with which you are working. The workspace also contains your SDK settings, software project files, and logs.

For convenience, SDK can be instructed to remember your last used workspace. When you enable this option, you will not be prompted to choose a workspace every time you start SDK. You can always switch your current workspace within SDK.

**IMPORTANT NOTE**: It is not safe to copy or move workspaces from one location to another. If you want to copy or move software components in your workspace, you must use the SDK Export/Import functions. This will be covered in Lab 8.

***Hardware Platform***

As you learned in Lab 1, the Hardware Platform Specification archive (hdf) captures all the information and files from a Xilinx® Vivado hardware design that are required for a software developer to write, debug, and deploy software applications for that hardware.

Each Workspace will typically have one hardware platform. There may be multiple BSPs and software applications, but these will always be associated with a single hardware platform.

**Experiment 1 General Instruction:**

| |
|---|
| Launch SDK. Create a Workspace in the SDK_Workspace directory. Import the Zynq hardware platform reviewed during Lab 1. |

**Experiment 1 Step-by-Step Instructions:**

1. Launch SDK by selecting **Start → All Programs → Xilinx Design Tools → SDK 2016.2 → Xilinx SDK 2016.2**.

2. If you have the default settings, SDK will ask you which workspace that you would like to open. We will open a new workspace. If you do not get this view, then select **File → Switch Workspace → Other**. Browse to `C:\Speedway\ZynqSW\2016_2\SDK_Workspace` and click **OK** to select the folder, then **OK** again to close the *Workspace Launcher*.

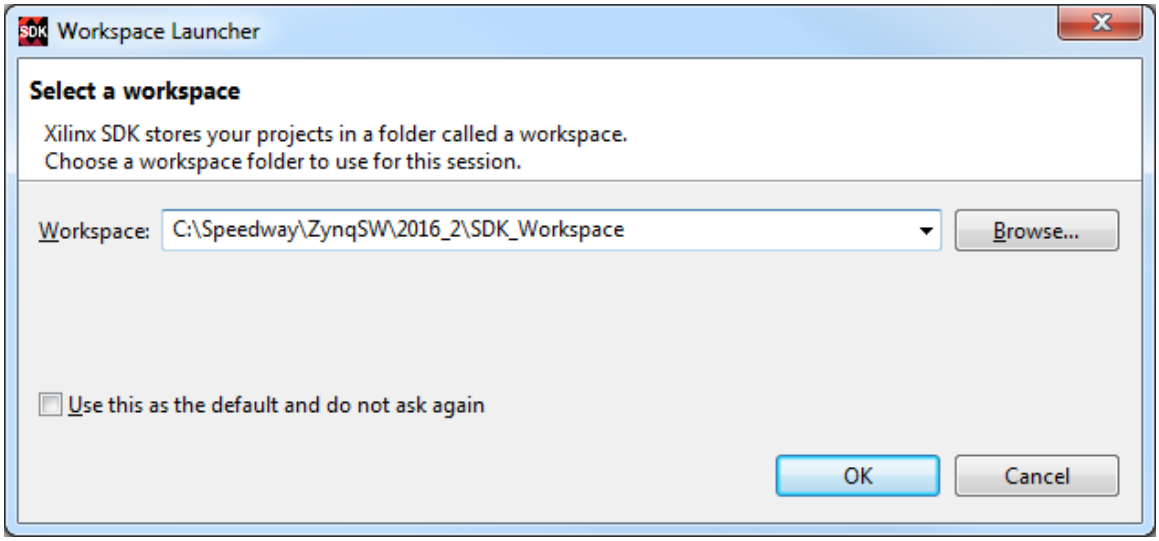


**Figure 1 – SDK Workspace, *Spaces Not Allowed***

The Xilinx Vivado 2016.2 tools provide for an absolute path relationship as well as a relative path definition to any custom IP repository that is used to create a hardware platform within Vivado. Those absolute and relative paths are included when exporting the hardware definition to SDK using the HDF file we explored in Lab 1. Matching one or both of these absolute/relative paths usually works okay for when a design engineer is working on a hardware platform in Vivado and then performs some software testing of the hardware using SDK on the same build machine. However, what happens when the hardware platform is handed off from a hardware engineer to a software engineer working on a different build machine?

The potential for both the absolute/relative paths to be mismatched between the different build machines now exists. This mismatch results in an invalid path condition which SDK detects and warns about when importing a hardware platform and becomes a problem when a custom driver (which comes with the custom IP) is needed for software development in SDK. This is important because custom drivers are identified by SDK when automatically building driver associations for hardware peripherals during the BSP creation process.

Fortunately this potential path discrepancy between build machines can be accounted for by using one of the following three approaches:

a) Modification of the **<REPOSITORIES>** property of the **Z_system.hwh** file contained within the HDF compressed archive in order to match the **ABSPATH** or **RELPATH** settings to a local folder containing the custom IP repository.

```
<REPOSITORIES>

<REPOSITORY ABSPATH="c:/Speedway/ZynqHW/2016_2/ip_repo/PWM_w_Int_1.0"
RELPATH="..\..\..\..\..\..\ip_repo\PWM_w_Int_1.0"/>

</REPOSITORIES>
```

b) Maintaining a custom IP repository on both PCs having the same absolute path. This is most likely already the case when using revision controlled custom IP within a department or company.

c) Specfiying a local repository within SDK allows for the custom IP drivers to be identified during BSP creation. This is the option that we will use for the remaining lab activities since it is a straightforward solution.

As a note, starting with the Vivado 2015.1 tools, any custom IP repository used to create a hardware platform is exported along with the HDF file which eliminates the need to worry about the potential for invalid paths to custom IP repositories when importing that hardware into SDK on another build machine.

Before the Hardware Platform is imported, now is a good time to make sure that SDK is configured with the a Local Repository. A local respository gives SDK access to local resource files which range from custom IP, custom drivers, custom BSPs, custom libraries, and even more.

The files provided with these lab activies include a **Support_Documents\sdk_repo** folder which we will use as a local repository containing the the custom Programmable Logic IP that gets created in the Zynq Developing Zynq®-7000 All Programmable SoC Hardware SpeedWay course as well as the AMP Standalone BSP both of which get used in later lab activities.

Next, the Hardware Platform will be imported. The hardware platform for this lab was previously designed and built for the Zynq 7010 SoC on the Avnet MicroZed and Zedboard. ***This hardware design process is not covered during this software-focused Speedway.*** To learn about this topic, please attend the Avnet Course *Developing Zynq Hardware*.

When using Windows, a Firewall security alert may appear, just select "allow access".

3. Select **File → New → Other**.

4. Expand the *Xilinx* item, and select **Hardware Platform Specification**. Click **Next >**.

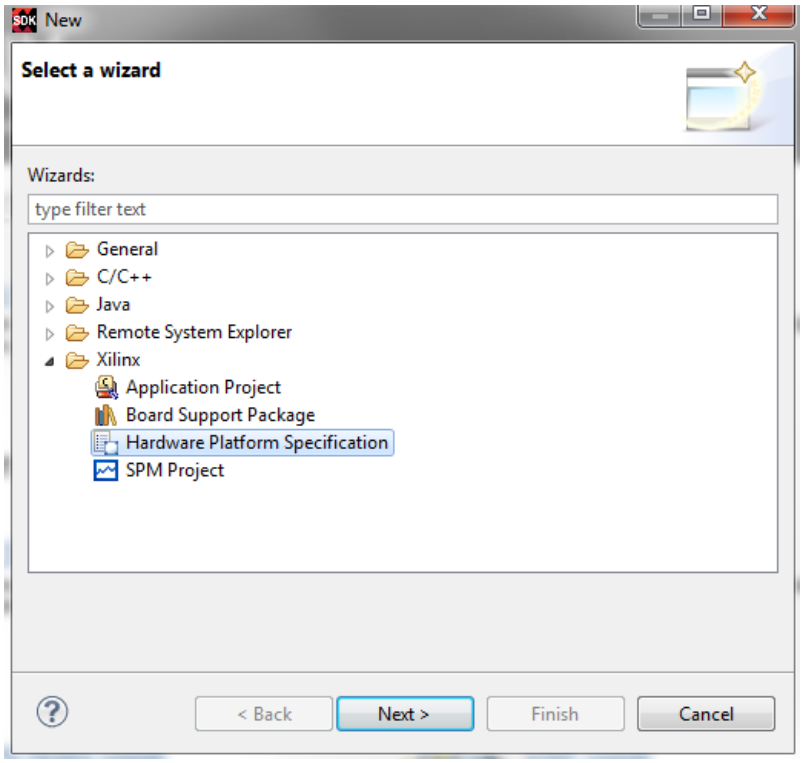


**Figure 2 – Creating a New Hardware Platform**

5. Insert **ZynqHW** for the *Project name*. Leave the *Use default location* checked.

6. Under *Target Hardware Specification*, click **Browse** and select the `Z_system_wrapper.hdf` file located in the Vivado project folder:

For MicroZed 7010

**C:\Speedway\ZynqSW\2016_2\ZynqDesign\MZ\7010\ZynqDesign.sdk**

For MicroZed 7020

**C:\Speedway\ZynqSW\2016_2\ZynqDesign\MZ\7020\ZynqDesign.sdk**

For PicoZed 7010

**C:\Speedway\ZynqSW\2016_2\ZynqDesign\PZ\7010\ZynqDesign.sdk**

For PicoZed 7015

**C:\Speedway\ZynqSW\2016_2\ZynqDesign\PZ\7015\ZynqDesign.sdk**

For PicoZed 7020

**C:\Speedway\ZynqSW\2016_2\ZynqDesign\PZ\7020\ZynqDesign.sdk**

For PicoZed 7030

**C:\Speedway\ZynqSW\2016_2\ZynqDesign\PZ\7030\ZynqDesign.sdk**

For ZedBoard

**C:\Speedway\ZynqSW\2016_2\ZynqDesign\ZB\ZynqDesign.sdk**

7. Click **Open** and **Finish**.

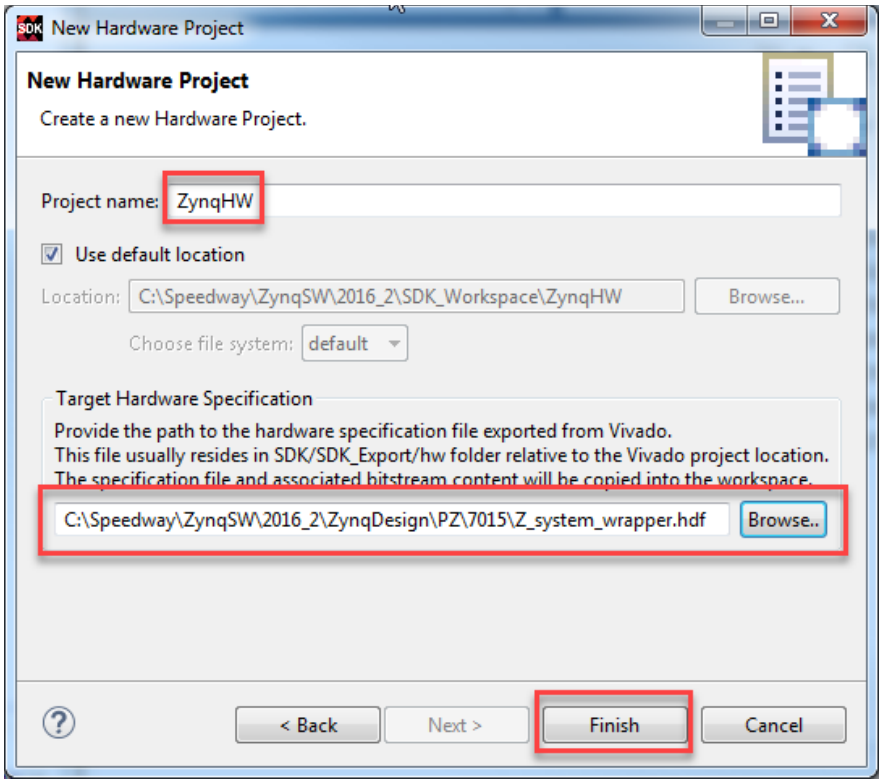


**Figure 3 – Create New Hardware Project**

8. Close the *Welcome* screen if it is still open.

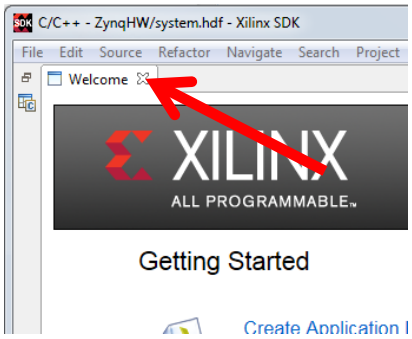


**Figure 4 – close the Welcome Screen**

9. Notice the PS7 Zynq hardware platform is now visible in the *Project Explorer*.

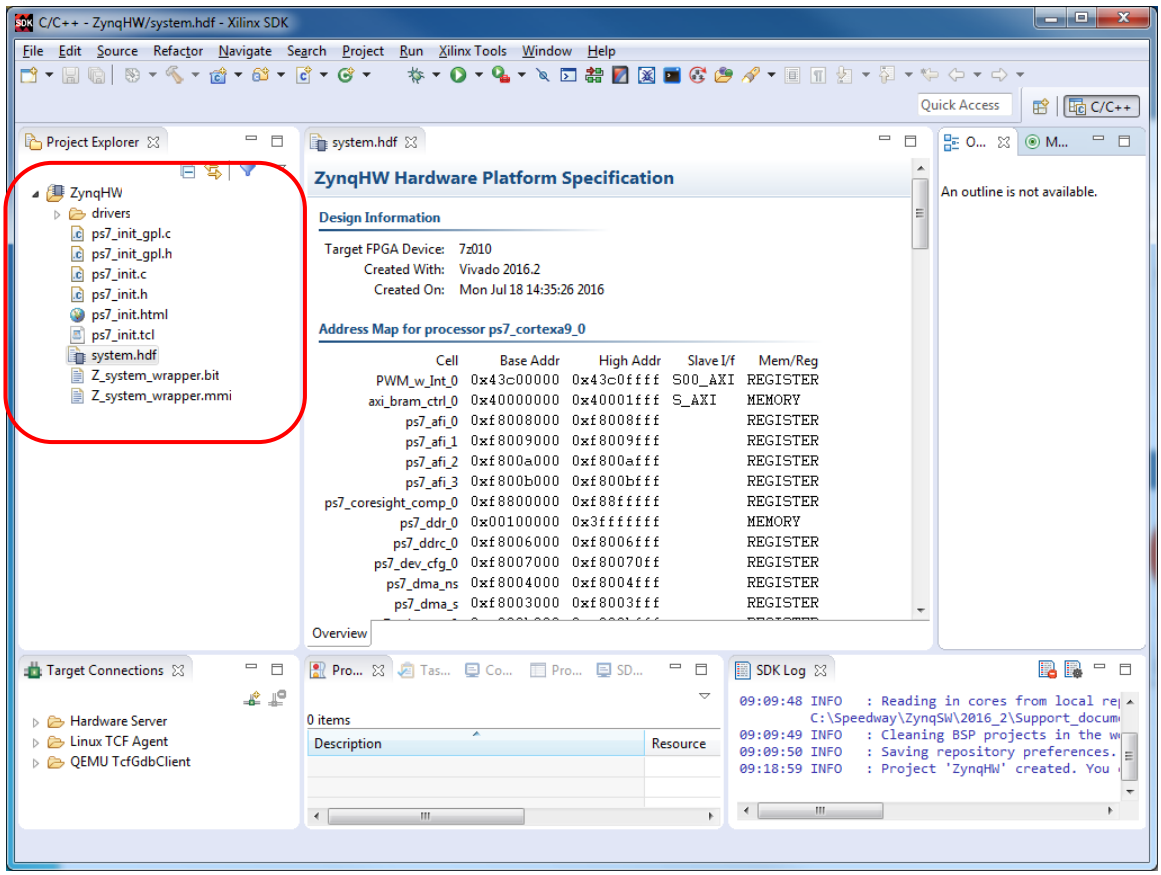


**Figure 5 – Hardware Platform Imported and Ready for Use**

## *Questions:*

***Answer the following questions:***

- *What is the purpose of the SDK Workspace?*

_____

_____

_____

- *When you need to share an SDK Workspace, is it a good idea to simply zip it up and send it over?*

_____

## Experiment 2: Examine the Hardware Platform

SDK is now ready for software development, but first we will review the hardware platform peripherals. This is roughly equivalent to reviewing the datasheet for a standard, off-the-shelf processor. The hardware platform specification provides many different important pieces of information, including:

- Peripheral set
- Address map
- Datasheets to peripherals
- System block diagram

**Experiment 2 General Instruction:**

| |
|---|
| Examine the hardware platform. Identify the peripherals included in this hardware platform. Review the peripheral address map. Open the Zynq datasheet. Review the external port list. Determine the system clock speed. |

**Experiment 2 Step-by-Step Instructions:**

1. The ***ZynqHW*** hardware platform is now visible under the Project Explorer. If ever not visible, the Project Explorer view can be restored by selecting **Window → Show View → Project Explorer**, or by resetting the entire perspective view with **Window → Reset Perspective**. You should recognize all of these files from the discussion in Lab 1.

2. The *Hardware Platform Specification* is already open in the working window since the hardware platform was just imported. To open this at other times, double-click on **system.hdf** in Project Explorer under ZynqHW.

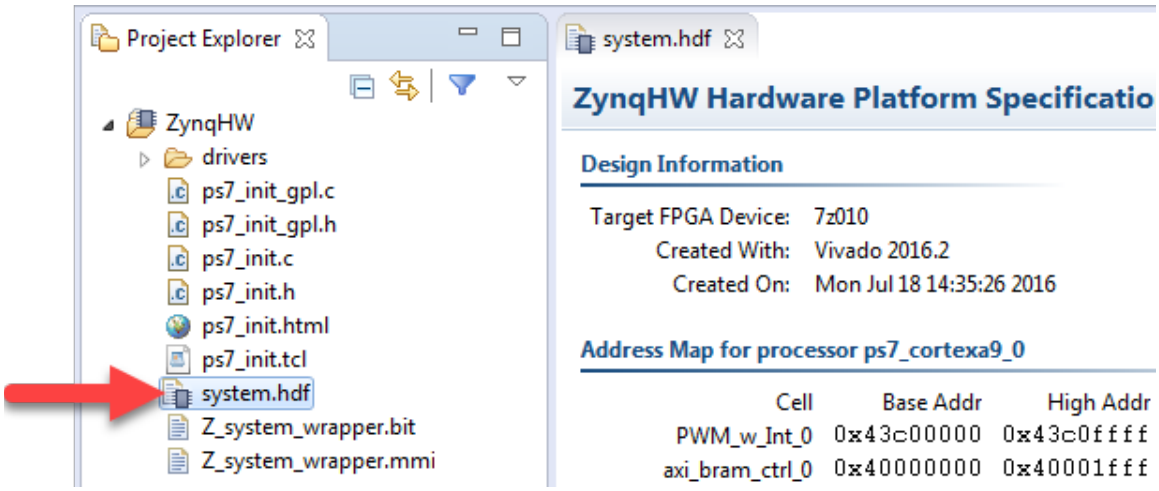


**Figure 6 – Hardware Platform Specification**

Aside from the basic design information section, the information is organized into three sections.

a. Address Map for processor ps7_cortexa9_0
b. Address Map for processor ps7_cortexa9_1
c. IP blocks present in the design

3. Under *Design Information*, note the Target FPGA Device is identified. The version of Vivado that created the hardware platform is also identified and time stamp of the design

4. Use these sections to answer the following questions.

## *Questions:*

***Answer the following question:***

- *What is address range for the DDR3 (ps7_ddr_0)?*

_____

- *How large is the QSPI Flash in the system? Note that peripheral ps7_qspi_0 is the QSPI controller while ps7_qspi_linear_0 is the Flash space.*

_____

- *What is the base address for the PWM Controller with Interrupt (PWM_w_Int_0)?*

_____

- *What is the version of the ARM Processor Core (ps7_cortexa9)?*

_____

5. Browse through system.hdf in SDK, specifically the ***IP blocks present in the design***. The first column represents the name of the IP in this design. The second column provides the name of the IP. We will focus on the second column and the name of the IP.

6. If you are unfamiliar with any of the peripherals in the design, it would be helpful to access a datasheet. Depending on the type of IP, you may find the IP datasheet in one of three different locations.

    a. Xilinx Processing System IP – the datasheet for these peripherals is found in the Zynq Technical Reference Manual, a copy of which is provided in the `C:\Speedway\ZynqSW\2016_2\Support_Documents` folder and in DocNav.

    b. Xilinx Programmable Logic IP – these are peripherals that are built in the PL fabric. The datasheets for this IP may be found online at www.xilinx.com, or you can make use of the Xilinx Documentation Navigator that is installed with the tools.

    c. Custom Programmable Logic IP – datasheets for custom IP must be obtained from the designer.

7. Use your PDF Reader to open the Zynq Technical Reference Manual (`ug585-Zynq-7000-TRM.pdf`) located in the following folder:

    `C:\Speedway\ZynqSW\2016_2\Support_Documents`

8. Looking at the IP names, choose any of the names that start ps7_*. These are all the Xilinx PS IP. For this document, the ps7_uart is used as an example.

9. In the TRM, search or browse to the UART section. This section explains the hardware IP for the UART Controller. Similarly, you could look in the TRM for descriptions of other PS7_* hardware.

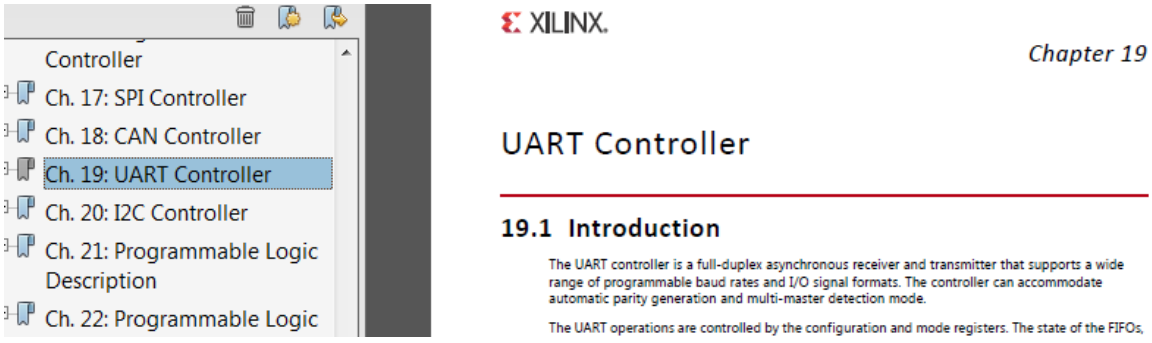


**Figure 7 – Documentation on UART Hardware Peripheral**

10. The axi_bram_ctrl peripheral in this design is one that is provided by Xilinx. As a piece of IP for the PL, it has its own Product Guide. The Product Guide is PG078 (axi_bram_ctrl_v4.0_Product_Guide). If you have DocNav installed then search for PG078 within DocNav

The axi_bram_ctrl_v4.0_Product_Guide is provided for your reference in the `Support_Documents` folder.

11. You can also search www.xilinx.com directly for "axi_bram_ctrl." Just be cautious in selecting the correct version. Notice the system.xml lists this IP as v4.0.

12. The last type of IP is Custom IP. This one is more difficult as documentation for custom IP is dependent on whoever created the IP. In this design, the PWM_w_Int_0 is a piece of custom IP. You would have to request the documentation from the designer. For this course, the hardware platform and IP is fully documented and explained in the *Developing Zynq Hardware* course.

13. The Vivado tools will also create a graphical block diagram of the hardware design. It may be beneficial to get the hardware engineer to provide a copy of this block diagram.

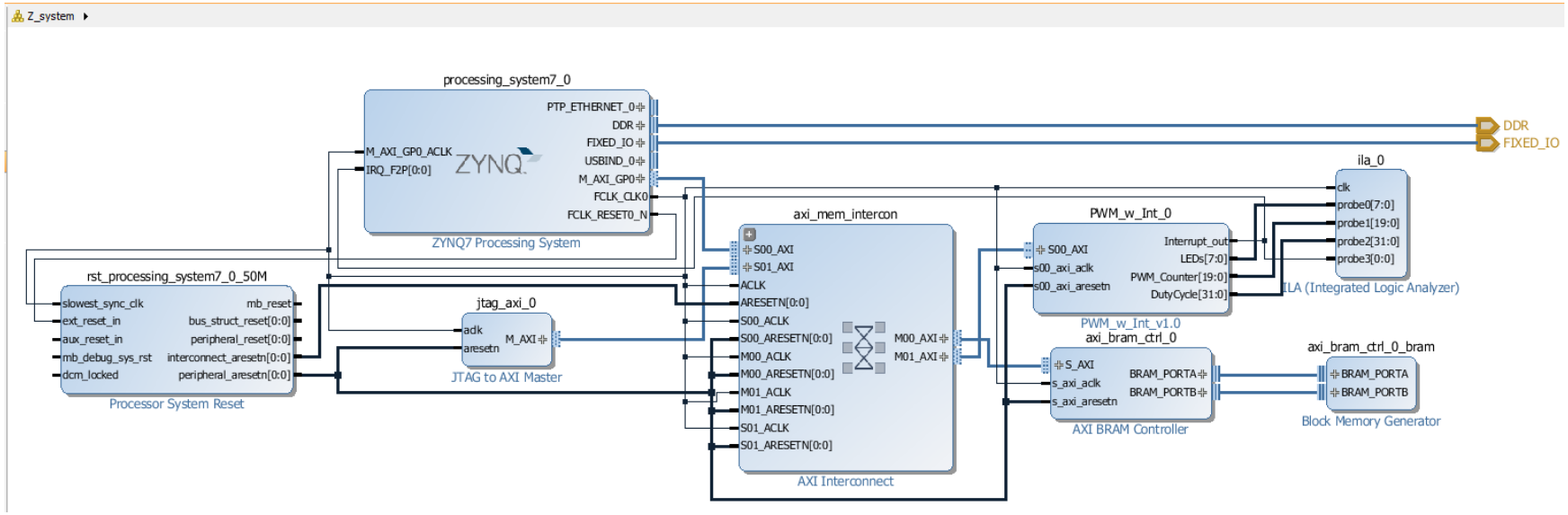


**Figure 8 – Hardware System Block Diagram**

***Question:***

***Answer the following question:***

- *Where will you find the hardware datasheet for the ps7* peripherals?*

_____

## Exploring Further

If you have more time and would like to investigate more…

- Review the *Cheat Sheets* available under SDK's **Help** menu.

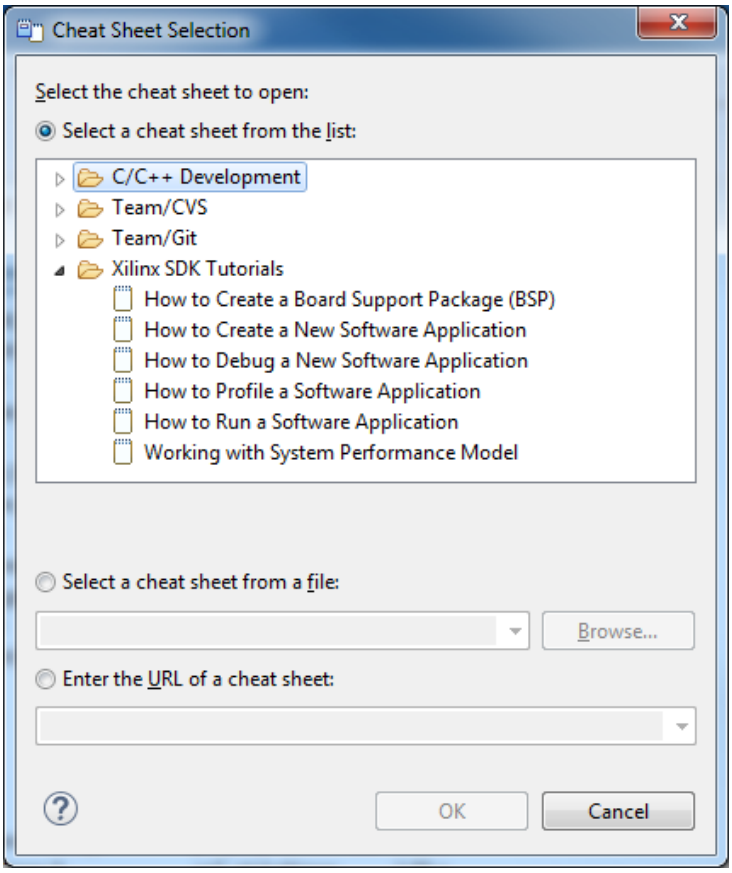


**Figure 9 – Cheat Sheets Are Tutorials**

This concludes Lab 2.

## Revision History

| Date | Version | Revision |
|---|---|---|
| 12 Nov 2013 | 01 | Initial release |
| 22 Nov 2013 | 02 | Revisions after pilot |
| 01 May 14 | 03 | MicroZed.org Training Course Release |
| 28 Oct 14 | 04 | Update to 2014.3 |
| 11 Dec 14 | 05 | Update to 2014.4 |
| 05 Jan 15 | 06 | Minor typo fix |
| 18 Mar 15 | 07 | Finalize SDK 2014.4 |
| Oct 15 | 08 | Updated to 2015.2 |
| Aug 16 | 09 | Updated to 2016.2 |

## Resources

www.microzed.org

www.picozed.org

www.zedboard.org

www.xilinx.com/zynq

www.xilinx.com/sdk

www.xilinx.com/vivado

www.xilinx.com/support/documentation/sw_manuals/ug949-vivado-design-methodology.pdf

www.xilinx.com/support/documentation/sw_manuals/ug1046-ultrafast-design-methodology-guide.pdf

# Answers

## Experiment 1

- *What is the purpose of the SDK Workspace?*

The Workspace holds a collection of projects related to your software application development. It will typically contain one hardware platform and one or more BSPs and software applications. The workspace contains your SDK settings, software sources, and logs.

- *When you need to share an SDK Workspace, is it a good idea to simply zip it up and send it over?*

NO! In Lab 8, we'll show you the proper way.

## Experiment 2

- *What is address range for the DDR3 (ps7_ddr_0)?*

0x0010_0000 to 0x1fff_ffff

- *How large is the QSPI Flash in the system? Note that peripheral ps7_qspi_0 is the QSPI controller while ps7_qspi_linear_0 is the Flash space.*

0xfcff_ffff – 0xfc00_0000 + 1 = 0x1000000 = 16 MB = 128 Mb

- *What is the base address for the PWM Controller with Interrupt (PWM_w_Int_0)?*

0x43c0_0000

- *What is the version of the ARM Processor Core (ps7_cortexa9)?*

5.2

- *Where will you find the hardware datasheet for the ps7* peripherals?*

The Zynq TRM